Express Mail Label No.: EV399912834US
Utility Patent Application
Atty. Docket No.: **SDV-001A**

MANAGING BURSTS OF DATA

Cross Reference to Related Applications

[0001] The present application claims the benefit of and priority to United States provisional application Serial No. 60/431,407, filed December 6, 2002, entitled "Method For Improving Memory Read Efficiency and Arithmetic Coding Speed," the entire disclosure of which is herein incorporated by reference.

Technical Field

[0002] The invention relates to managing bursts of data. In particular, an aspect of the invention relates to storing memory data multiple times within a memory with different burst alignments such that the required number of bursts may be reduced for the worst case.

Background of the Invention

[0003] Video decoders include a function commonly referred to as "motion compensation." This function is necessary to allow the decoder to process numerous different video compression standards, including but not necessarily limited to: MPEG-1, MPEG-2, MPEG-4, H.263, H.264, and H.26l. More specifically, motion compensation includes a process of copying a two-dimensional block of image data from a previously decoded reference frame to the frame currently being decoded. The location of the reference block relative to the current position in the current frame is specified by "motion vectors" included within the input code stream. Motion compensation allows for

a compact specification of the data whenever the video stream is well modeled by translational motion.

[0004] Typically, reference frames that are used for motion compensation are stored in a relatively large memory (typically DRAM). To improve general performance, DRAMs are generally accessed in bursts of data (usually 2, 4, 8, or 16 data words in a burst). For memory data interfaces that have a word size of 4 bytes and are 32 bits wide and a burst size of 2 words, each burst accesses 8 bytes of data (2 words x 4 bytes per word). However, bursts can only access data aligned to burst boundaries, and therefore a burst of N words must be aligned to an address integrally divisible by N.

Summary of the Invention

[0005] In general, the invention relates to managing bursts of data. Aspects of the invention relate to methods of storing and retrieving data that reduces the number of bursts needed to read data aligned in the worst case manner with respect to burst boundaries.

[0006] In at least one aspect, the invention relates to a method of storing data. The method includes storing data in a machine readable memory device a first time at a first memory address, where the machine readable memory device has one or more burst boundaries and the first memory address has a first alignment with respect to the burst boundaries, and storing the data in the machine readable memory device a second time at a second memory address, where the second memory address has a second alignment with respect to the burst boundaries.

[0007] In at least some embodiments, the data represents a reference frame for use in a video decoder, which in some embodiments may be an h.264 codec. The machine readable memory device can be volatile memory such as static random access memory or in some embodiments dynamic random access memory. In some embodiments, the

machine readable memory device may be non-volatile memory such as read-only memory.

[0008] In at least some embodiments, the data may be stored in the machine readable memory device a third time at a third memory address, where the third memory address has a third alignment with respect to the burst boundaries.

[0009] In another aspect, the invention relates to a method of retrieving data from a machine readable memory device. The method includes determining a set of desired bytes of data, the set of desired bytes of data having been previously stored in a machine readable memory device at two or more memory addresses, the memory device having at least one burst boundary, and each memory address having a different alignment with respect to the at least one burst boundary. The method further includes retrieving the desired bytes of data from a preferred the memory address, the preferred memory address being aligned with the at least one burst boundary such that the number of bursts necessary to read the desired bytes from the preferred memory address is fewer than the number of bursts necessary to read the desired bytes from the other memory addresses.

[0010] While particularly useful in the field of encoded video data, these methods are not limited to that specific application, and can be used in similar applications where data is stored in and read from memory devices.

Brief Description of the Drawings

[0011] In the drawings, like reference characters generally refer to the same elements throughout the different views. In addition, the drawings are not necessarily to scale, emphasis instead generally being placed upon illustrating the principles of the invention.

[0012] FIG. 1 illustrates storing data at one memory address.

[0013] FIG. 2 illustrates another example of storing data at one memory address.

[0014] FIG. 3 illustrates storing data at two memory addresses.

[0015] FIG. 4 illustrates another example of storing data at two memory addresses.

Detailed Description

[0016] Reference blocks used for motion compensation consist of 2 dimensions of image data which are often small relative to the burst size of a memory device such as DRAM. Video data is stored in raster order – i.e. it is scanned on a row by row basis with the upper left pixel being stored at the lowest address of the picture memory and the lower right pixel at the highest address in the picture memory. For example, a 13 pixel by 13 line block of a single image component may be broken into 13 reads (1 read for each line). Where one component of one pixel equals one byte of data, 13 bytes can be read from each line for each image component. Where the memory data interface to the computing device is 32 bits wide (a word size of 4 bytes) and the burst size is 2 words, then each burst accesses 8 bytes of data (2 words x 4 bytes per word).

[0017] Therefore, even though the data for each line of the reference block may be read from DRAM in either two or three bursts, the worst case burst alignment must be assumed to account for the alignments requiring a three burst read. This results in reduced read efficiency and slower processing. In cases where the video data is of a high resolution, the slower processing may limit the quality of the data the system can process.

[0018] Referring to FIG.1, a set of memory data bytes represented by the letters A – Y (105a) resides at a given memory address within a machine readable memory such as DRAM. Further, a subset of the set of 13 memory data bytes D – P (110) represents a desired subset of the set of memory data bytes 105a in that they contain information requested by a computing device (not shown). However, bursts can only access data aligned to burst boundaries 115 (i.e. a burst of N words must be aligned to an address integrally divisible by N). Therefore, only 2 bursts of 8 bytes are required to read the 13 desired bytes (110) – burst 1 reads bytes A – H and burst 2 reads bytes I – P. In total, bytes A – P are read in order to obtain the data in bytes D – P. This yields an efficiency

of 81% on DRAM data cycles (reading 16 bytes to get the 13 desired bytes.)

[0019] Referring to FIG. 2, if the desired data 110 is stored at data bytes G - S, the alignment of the memory data bytes 105a and the burst boundaries 115 require 3 bursts of 8 bytes to get the desired data 110. Thus, 24 bytes of data are read (A – X) to retrieve 12 bytes of desired data G – S 110. This yields an 54% efficiency on DRAM data cycles (reading 24 bytes to get the 13 desired bytes.) When designing to the worst case, the lowest efficiency number must be used because it is possible that the required data will always have the worst alignment as illustrated in FIG. 2. When designing to the average case, the reduced efficiency of the FIG. 2 case will degrade the average DRAM read efficiency achieved.

[0020] To improve the read efficiency, memory data is stored multiple times within the memory with different burst alignments such that the required number of bursts is reduced for the worst case alignment. As a result, both the worst-case and average efficiency improve. Accesses within a single burst are highly efficient (1 word per clock cycle with single data rate (SDR) DRAM, 2 words per clock cycle with double data rate (DDR) DRAM).

[0021] FIG. 3 illustrates an example embodiment in which the same set of data is stored twice (110a and 110b). To retrieve the desired bytes of data D – P (105b), the first set of data 110a is used. To do so, only two bursts of data are needed (burst 1 and burst 2) and the read efficiency is 81% (13 desired bytes / 16 read bytes). However, as illustrated in FIG. 4, if the bytes G – S 105c are requested, the data is read from the second set of data 110b. Reading the desired data 105c from the first data segment would require three bursts, while reading the desired data 110c from data segment 2 requires only two bursts. With the data duplicated twice, all accesses of 13 bytes or less are now guaranteed to only require 2 bursts, leading to a worst case (and average) efficiency of 81%.

[0022] In one embodiment, the segment of data with the better alignment with respect to the burst boundaries is determined using combinatorial logic. For example, let *A*

represent the address of the first byte sequence. If (A modulo 8) ≤ 3, the first data segment is used. If (A modulo 8) > 3, the second data segment is used.

[0023] Another embodiment uses duplication factors greater than 2. By duplicating the data more times, more alignments can be stored, which is highly useful when different desired data lengths or different burst sizes are required. As a result, at least one implementation relies on a tradeoff of memory usage and write timing for reading efficiency.

[0024] In one particular instance, an implementation can be applied to any video decoder and related encoder requiring motion compensation. In one embodiment, H.26l codecs are used as decoders. In general, the technique can improve read efficiency significantly in any burst oriented memory subsystem (video codec or other) that has reads which required a relatively small number of bursts.

[0025] The methods described above may be implemented using one or more data processing devices. In some embodiments, the data processing devices may implement the functionality of the present invention in hardware, using, for example, a computer chip. The data processing device may receive signals in analog or digital form. In other embodiments, the data processing device may implement the functionality of the present invention as software on a general purpose computer, video display device, or other electronic device. In such an embodiment, the program may be written in any one of a number of programming languages, such as FORTRAN, PASCAL, C, C++, C#, Tcl, or BASIC. Further, the program can be written in a script, macro, or functionality embedded in commercially available software, such as EXCEL or VISUAL BASIC.

[0026] Additionally, the software could be implemented in an assembly language directed to a microprocessor resident on a video display device, computer or other electronic device. For example, the software can be implemented in Intel 80x86 assembly language if it is configured to run on an IBM PC or PC clone. The software may be embedded on an article of manufacture including, but not limited to, "machine-

readable program means" such as a floppy disk, a hard disk, an optical disk, a magnetic tape, a PROM, an EPROM, ROM, or CD-ROM.

[0027] Variations, modifications, and other implementations of what is described herein will occur to those of ordinary skill in the art without departing from the spirit and the scope of the invention as claimed. Accordingly, the invention is to be defined not by the preceding illustrative description but instead by the spirit and scope of the following claims.

What is claimed is: